

Geospatial Optimization Problems

Paulo Shakarian
Network Science Center
Dept. of Electrical Engineering and
Computer Science
U.S. Military Academy
West Point, NY 10996
paulo[at]shakarian.net

V.S. Subrahmanian
Dept. of Computer Science
University of Maryland
College Park MD
vs[at]cs.umd.edu

Abstract—There are numerous applications which require the ability to take certain actions (e.g. distribute money, medicines, people etc.) over a geographic region. A disaster relief organization must allocate people and supplies to parts of a region after a disaster. A public health organization must allocate limited vaccine to people across a region. In both cases, the organization is trying to optimize something (e.g. minimize expected number of people with a disease). We introduce “geospatial optimization problems” (GOPs) where an organization has limited resources and budget to take actions in a geographic area. The actions result in one or more properties changing for one or more locations. There are also certain constraints on the combinations of actions that can be taken. We study two types of GOPs - goal-based and benefit-maximizing (GBGOP and BMGOP respectively). A GBGOP ensures that certain properties must be true at specified locations after the actions are taken while a BMGOP optimizes a linear benefit function. We show both problems to be NP-hard (with membership in NP for the associated decision problems). Additionally, we prove limits on approximation for both problems. We present integer programs for both GOPs that provide exact solutions. We also correctly reduce the number of variables in for the GBGOP integer constraints. For BMGOP, we present the **BMGOP-Compute** algorithm that runs in PTIME and provides a reasonable approximation guarantee in most cases.

I. INTRODUCTION

As geo-located social network data becomes more common with sites such as FourSquare¹ and programs such as RealityMining², it becomes desirable to reason about such data. There are numerous applications which require the ability to take certain actions (e.g. distribute money, medicines, people etc.) over a geographic region. For instance, a disaster relief organization must allocate people and supplies in a region after a disaster. A public health organization needs to allocate limited vaccine stocks to people across the region. A government needs to allocate funds for education or unemployment training across a region. However, allocating any resource will cause certain effects - some desirable, some not - based on the network connections among geographic locations. In this paper we present a formal framework that allows reasoning about such geo-located data in order to answer certain queries where we have some desired goal to achieve as the result of

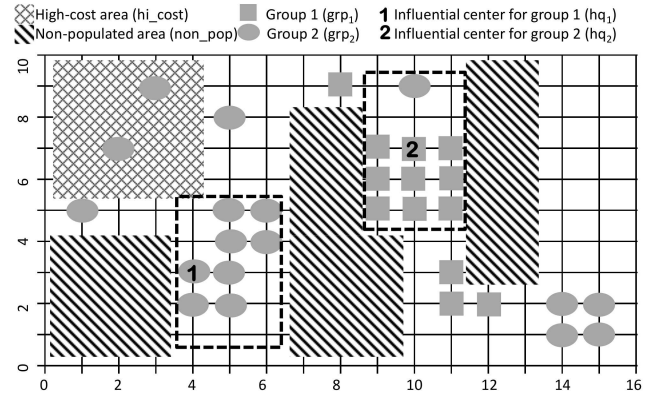


Fig. 1. Locations in a district - contingency groups and unpopulated areas.

our geographically-based resource allocation - **all the while considering the complex interactions among locations.**

Figure 1 shows a 2-dimensional map of a region. A political candidate can only make so many campaign stops and public appeals. We assume that a map \mathcal{M} is discrete (this is a common assumption in most GIS systems) and has coordinates drawn from $[0, \dots, M] \times [0, \dots, N]$ where the bottom left corner of the map is the point $(0, 0)$. The candidate wants to identify the best places to campaign or make public appeals to maximize his exposure. Additionally, the map shows unpopulated areas, areas where campaigning costs are high, and areas dominated by one of two constituent groups. All of these factors may affect the set of locations the candidate selects to optimize his exposure.

In this paper, we introduce *geographic optimization problems* or GOPs that capture and solve problems such as those mentioned above. This framework allows one to more prudently position resources in a manner to achieve a goal while considering the complex interactions between locations (that may be modeled as a network). The organization and contribution of the paper is as follows. Section II formally defines GOPs - specifically we introduce goal-based and benefit-maximizing GOPs (GBGOP and BMGOP respectively). Section III shows that both GBGOP and BMGOP are NP-hard (with the associated decision problems in the complexity class

¹<https://foursquare.com/>

²<http://realitycommons.media.mit.edu/>

NP). Additionally, we prove non-trivial theoretical limits on approximation: if GBGOP were to be approximated within the logarithm of the input then NP would have a slightly super-polynomial oracle. BMGOP cannot be approximated within a guaranteed factor greater than 0.63 unless P=NP. Section IV presents integer programs to solve both GBGOP and BMGOP using an IP solver like CPLEX. In Section V, we show how to correctly reduce the number of variables in the integer constraints for GBGOP. We then develop the BMGOP-Compute algorithm in Section VI that can quickly approximate a BMGOP in polynomial time and provides an approximation guarantee.

II. GOPS FORMALIZED

Throughout this paper, we assume that $\mathcal{M} = [0, \dots, M] \times [0, \dots, N]$ is an arbitrary, but fixed “map”. We define a logical language \mathcal{L} whose constant symbols are members of \mathcal{M} and that has an infinite set \mathcal{L}_{var} of variable symbols disjoint from \mathcal{M} . \mathcal{L} has a set $\mathcal{G} = \{g_1, \dots, g_n\}$ of unary predicate symbols. As usual, a term is either a constant symbol or variable symbol. If t is a term, then $g_i(t)$ is an *atom*. If t is a constant, then $g_i(t)$ is *ground*. Intuitively, if $p \in \mathcal{M}$, then $g_i(p)$ says that point p has property g_i . We use $B_{\mathcal{L}}$ to denote the set of all ground atoms. Well-formed formulas (wffs) are defined in the usual way. (i) Every atom is a wff. (ii) If F, G are wffs, then so are $F \wedge G, F \vee G, \neg F$ are all wffs.

Example 2.1: Consider the map \mathcal{M}_{cpgn} in Figure 1 with predicates $\mathcal{G} = \{hi_cost, non_pop, grp_1, grp_2, hq_1, hq_2\}$. The predicate *exposure* not depicted in the figure corresponds to a candidate receiving exposure in a certain area. $hi_cost((1, 9)), hq_1((4, 3)), non_pop((8, 1))$, and $grp_2((5, 8))$ are all examples of ground atoms.

A *state* is any subset of $B_{\mathcal{L}}$. We use \mathbf{S} to denote the set of all states. Satisfaction of formulas is defined in the obvious way. State s satisfies a ground atom A , denoted $s \models A$, iff $A \in s$. $s \models F \vee G$ iff $s \models F$ or $s \models G$. $s \models F \wedge G$ iff $s \models F$ and $s \models G$. $s \models \neg F$ iff s does not satisfy F .

Example 2.2: The shading shown in Figure 1 defines a state. For example, $hi_cost((1, 9)) \in s_{cpgn}$ while $exposure((1, 9)) \notin s_{cpgn}$.

An action maps points to sets of ground atoms.

Definition 2.1 (Action): An action is a mapping $a : \mathcal{M} \rightarrow 2^{B_{\mathcal{L}}}$. We use \mathcal{A} to denote the set of actions. An action-point pair is any member of $\mathcal{A} \times \mathcal{M}$.

An action-point pair (a, p) is executed if action a takes place at point p . Thus, one can think of (a, p) as saying that action a occurs at point p . The *result of executing a set SOL of action-point pairs in state s_0* is denoted $appl(SOL, s_0)$ and is the set $(s_0 \cup \{a(p) \mid (a, p) \in SOL\})$.

Example 2.3: Continuing with example 2.6, our candidate has actions $\mathcal{A}_{cpgn} = \{nor, appeal_1, appeal_2\}$ where *nor* refers to a normal campaign stop and $appeal_1, appeal_2$ refer to public appeals to constituent groups 1 and 2 respectively. The actions map to ground atoms as follows.

$$\begin{aligned} nor(p) &= \{exposure(p') \mid \neg non_pop(p') \wedge d(p, p') \leq 1\} \\ appeal_i(p) &= \{exposure(p') \mid hq_i(p) \wedge grp_i(p')\} \end{aligned}$$

The first action says that when a normal campaign stop is made at point p and p' is a populated place one distance unit or less from p , then the candidate has exposure at place p' as well. The second action says that if the candidate makes an appeal (action) at point p and p is the headquarters of interest group grp_i , then the candidate has obtained exposure in all places associated with interest group grp_i .

Definition 2.2 (Cost Function): A **cost function**, $\mathbf{C} : \mathcal{A} \times \mathcal{M} \rightarrow [0, 1]$.

Throughout this paper, we assume the cost function is arbitrary but fixed and can be computed in constant time. We also assume that if $\mathcal{A} \times \mathcal{M} = \{(a_1, p_1), \dots, (a_m, p_m)\}$, then c_i is used to denote $\mathbf{C}(a_i, p_i)$.

Example 2.4: The cost function for our example is $\mathbf{C}_{cpgn}^{(s)}$ and is defined (based on some state s) as follows: $\mathbf{C}_{cpgn}^{(s)}(a, p) = 1$ if $hi_cost(p) \in s$ and 0.5 otherwise.

We also assume the existence of a set of integrity constraints IC that specify that certain actions cannot be jointly taken if some conditions hold w.r.t. the state — such constraints were defined before by [1].

Definition 2.3 (Integrity Constraint): If Φ is a set of action-point pairs and χ is a wff, then $\Phi \leftrightarrow \chi$ is an **integrity constraint**.

When $\Phi \leftrightarrow \chi$ is ground, this says that if χ is true, then only one action-point pair in Φ may be executed. Formally, suppose s is a state and Φ' is a set of action-point pairs and $\Phi \leftrightarrow \chi$ is ground. $(s, \Phi') \models \Phi \leftrightarrow \chi$ iff either $s \not\models \chi$ or $s \models \chi$ and $|\Phi \cap \Phi'| \leq 1$. (s, Φ') satisfies an integrity constraint iff it satisfies all ground instances of it. $(s, \Phi') \models IC$ where IC is a set of integrity constraints iff (s, Φ') satisfies every constraint in that set. Given a state s and set IC of integrity constraints, we use IC_s to denote the set of all ground instances of integrity constraints in IC where the associated wff χ is satisfied by s^3 .

Example 2.5: Continuing Example 2.4, let IC_{cpgn} be $\{\{appeal_1((4, 3)), appeal_2((10, 7))\} \leftrightarrow \text{TRUE}\}$. This constraint says that an appeal can be made to either group 1 or group 2 at their center of influence, but not both — for instance, these two groups may have opposing views.

We now introduce the *goal-based geospatial optimization problem* (GBGOP). This problem takes as input a map \mathcal{M} , initial state s_0 , set of actions \mathcal{A} , cost function \mathbf{C} , integrity constraints IC , positive real number \mathbf{c} , and disjoint sets $\Theta_{in}, \Theta_{out} \subseteq B_{\mathcal{L}}$. Intuitively, \mathbf{c} restricts the total cost and Θ_{in} (resp. Θ_{out}) is a set of atoms that must be true (resp. false) after the actions are applied. Our optimality criteria for a GBGOP is to minimize the cardinality of the action-point pairs. A GBGOP can be viewed as an abductive inference problem (i.e. find a set of actions that lead to the current state) - where minimal cardinality is a common parsimony requirement.

Definition 2.4 (GBGOP Solution, Optimal Solution): A *solution* to a GBGOP $(\mathcal{M}, s_0, \mathcal{A}, \mathbf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$ is a set $SOL \subseteq \mathcal{A} \times \mathcal{M}$ such that: (i) $\sum_{(a_i, p_i) \in SOL} c_i \leq \mathbf{c}$,

³Formally, $IC_s = \{(\Phi \leftrightarrow \chi) \in IC \mid s \models \chi\}$

(ii) $(s_0, SOL) \models IC$, and (iii) $appl(s_0, SOL) \models \bigwedge_{A_i \in \Theta_{in}} A_i \wedge \bigwedge_{A_j \in \Theta_{out}} \neg A_j$.

A solution SOL is *optimal* iff there is no other solution SOL' such that $|SOL'| \leq |SOL|$.

Our next type of problem is a *benefit-maximizing geospatial optimization problem* (BMGOP) that also considers a benefit function, defined as follows.

Definition 2.5 (Benefit Function): The **benefit function**, $B: B_{\mathcal{L}} \rightarrow \mathbb{R}^+$ maps atoms to positive real numbers.

Example 2.6: In our running example, we use the benefit function B_{cpn} where $B_{cpn}(A) = 1$ if A has the form $exposure()$ and 0 otherwise.

As with cost, we assume the benefit function to be arbitrary but fixed and computable in constant time. We also assume that if $B_{\mathcal{L}} = \{A_1, \dots, A_n\}$, then $B(A_i)$ is denoted b_i . A BMGOP takes as input, \mathcal{M} , s_0 , \mathcal{A} , \mathbf{C} , IC , and \mathbf{c} - all defined the same as for a GBGOP. Additionally it takes benefit function B and natural number k . Here k is a bound on the number of actions the agent can take as we attempt to maximize benefit as an optimality criteria.

Definition 2.6 (BMGOP Solution, Optimal Solution):

A solution to a BMGOP $(\mathcal{M}, s_0, B, \mathcal{A}, \mathbf{C}, IC, k, \mathbf{c})$ is a set $SOL \subseteq \mathcal{A} \times \mathcal{M}$ such that: (i) $|SOL| \leq k$ and (ii) $\sum_{(a_i, p_i) \in SOL} c_i \leq \mathbf{c}$, and (iii) $(s_0, SOL) \models IC$.

A solution SOL is *optimal* iff there is no other solution SOL' such that $\sum_{A_i \in appl(SOL, s_0)} b_i < \sum_{A_i \in appl(SOL', s_0)} b_i$.

III. COMPLEXITY RESULTS

Here, we provide complexity results for GBGOPs and BMGOPs. First, we establish both as being at least NP-hard.

Theorem 1: Given GBGOP $(\mathcal{M}, s_0, \mathcal{A}, \mathbf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, finding an optimal solution $SOL \subseteq \mathcal{A} \times \mathcal{M}$ is NP-hard. This result holds even if for each $a \in \mathcal{A}$, $p \in \mathcal{M}$, it is the case that $\forall g'(p') \in a(p)$, $p' = p$ - i.e. each action only affects the point it is applied to.

Proof Sketch. We embed the known NP-hard problem of SET-COVER [2] which takes as input a set of n elements, S and a family of m subsets of S , $\mathcal{H} \equiv \{H_1, \dots, H_m\}$, and outputs $\mathcal{H}' \subseteq \mathcal{H}$ s.t. the union of the subsets covers all elements in S and \mathcal{H}' is of minimal cardinality. We encode this problem into a GBGOP as follows: we set $\mathcal{G} = \{g_1, \dots, g_n\}$ - each predicate in \mathcal{G} corresponds to an element in S , the map, \mathcal{M} consists of a single point, p , the actions $\mathcal{A} = \{a_1, \dots, a_m\}$ s.t. each action a_i corresponds to an element in \mathcal{H} and each is defined as follows: $a_i(p) = \bigcup_{x_j \in H_i} \{g_j(p)\}$. The cost function \mathbf{C} returns 1 for each action-point pair, $\Theta_{in} = \bigcup_{g_i \in \mathcal{G}} \{g_i(p)\}$, $\Theta_{out} = \emptyset$, and finally, we set $s_0 = \emptyset$, $IC = \emptyset$, $\mathbf{c} = n$. \square

Theorem 2: Given BMGOP $(\mathcal{M}, s_0, B, \mathcal{A}, \mathbf{C}, IC, k, \mathbf{c})$, finding an optimal solution $SOL \subseteq \mathcal{A}$ is NP-hard. This result holds even if for each $a \in \mathcal{A}$, $p \in \mathcal{M}$, it is the case that $\forall g'(p') \in a(p)$, $p' = p$ - i.e. each action only affects the point it is applied to.

Proof Sketch. We embed the known NP-hard problem of MAX-K-COVER [2] which takes as input a set of n elements,

S and a family of m subsets of S , $\mathcal{H} \equiv \{H_1, \dots, H_m\}$, and positive integer K and outputs $\leq K$ subsets from \mathcal{H} s.t. the union of the subsets covers a maximal number of elements in S . We encode this problem into a BMGOP as follows: we set $\mathcal{G} = \{g_1, \dots, g_n\}$ - each predicate in \mathcal{G} corresponds to an element in S , the map, \mathcal{M} consists of a single point, p , the function B returns 1 for each ground atom, the set $\mathcal{A} = \{a_1, \dots, a_m\}$ is set s.t. each action in \mathcal{A} corresponds to an element in \mathcal{H} and each a_i is defined as follows: $a_i(p) = \bigcup_{x_j \in H_i} \{g_j(p)\}$. The cost function \mathbf{C} returns 1 for each action-point pair, and finally, we set $s_0 = \emptyset$, $IC = \emptyset$, $k = K$, $\mathbf{c} = K$. \square

One may think that one can solve GOPs efficiently in practice by using fully polynomial time approximation schemes (FPTAS). However, by the nature of our constructions used in the NP-hardness results, this is not possible for either type of GOP under accepted theoretical assumptions.

Theorem 3: If for some $\epsilon > 0$, there is a PTIME algorithm to approximate GBGOP within $(1 - \epsilon) \cdot \ln(|\mathcal{A} \times \mathcal{M}|)$, then $NP \subset TIME(|\mathcal{A} \times \mathcal{M}|^{O(\lg \lg |\mathcal{A} \times \mathcal{M}|)})$ (NP has a slightly super-polynomial algorithm).

Follows from Theorem 1 and [2, Theorem 4.4]. \square

Theorem 4: Finding an optimal solution to BMGOP cannot be approximated in PTIME within a ratio of $\frac{e-1}{e} + \epsilon$ (approx. 0.63) for some $\epsilon > 0$ (where e is the inverse of the natural log) unless $P=NP$, even when $IC = \emptyset$.

Follows from Theorem 2 and [2, Theorem 5.3]. \square

Next, under some reasonable assumptions, the decision problems for GBGOP/BMGOP are in-NP.

Theorem 5: Given GBGOP $(\mathcal{M}, s_0, \mathcal{A}, \mathbf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, if the cost function and all actions $a \in \mathcal{A}$ can be polynomially computed, then determining if there is a solution SOL for the instance of the GBGOP s.t. for some real number k , $|SOL| \leq k$ is in-NP.

Theorem 6: Given BMGOP $(\mathcal{M}, s_0, B, \mathcal{A}, \mathbf{C}, IC, k, \mathbf{c})$, if the cost function, benefit function, and all actions $a \in \mathcal{A}$ can be polynomially computed, then determining if there is a solution SOL for the instance of the BMGOP s.t. for some real number val , $\sum_{A_i \in appl(SOL, s_0)} b_i \geq val$ is in-NP.

As stated earlier, a GBGOP may also be viewed as an abductive inference problem. Even though finding a solution (not necessarily optimal) to a GBGOP can trivially be conducted in PTIME⁴, counting the number of solutions is #P-complete. This counting problem is difficult to approximate.

Theorem 7: Counting the number of solutions to a GBGOP (under the assumptions of Theorem 5) is #P-complete.

Proof Sketch. The MONSAT problem [3] takes a set C of m clauses of K disjunct ed literals (no negation) over set L of atoms (size n) and outputs “yes” iff there is a subset of L that satisfies all clauses in C . We encode this into finding a GBGOP as follows: $\mathcal{G} = \{g_1, \dots, g_m\}$ - each predicate in \mathcal{G}

⁴Return the set $\{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}$

corresponds to an clause in C (predicate g_j corresponds with clause ϕ_j), \mathcal{M} consists of a single point, p , $\mathcal{A} = \{a_1, \dots, a_n\}$ - each action in \mathcal{A} corresponds to an element in L (action a_i corresponds with literal ℓ_i). Each a_i is defined as follows: $a_i(p) = \{g_j(p) | \{\ell_i\} \models \phi_j\}$, \mathbf{C} returns 1 for all action-point pairs, $s_0 = \emptyset$, $IC = \emptyset$, $\mathbf{c} = n$, $\Theta_{in} = \bigcup_{g_i \in \mathcal{G}} \{g_i(p)\}$, $\Theta_{out} = \emptyset$. Based on this PTIME reduction we show a 1-1 correspondence to MONSAT. Hence, we can parsimoniously reduce the counting version of MONSAT (number of solutions) to the counting version of GBGOP (number of solutions). As the counting version of MONSAT is #P-hard by [3], we have shown that #P-hardness of the counting version of GBGOP. As there is an obvious bound on the number of solutions to a GBGOP, and as the solutions are verifiable in PTIME, membership in #P follows. \square

Theorem 8: For $\epsilon > 0$, approximating the number of solutions to a GBGOP within a factor of $2^{|\mathcal{A} \times \mathcal{M}|^{1-\epsilon}}$ is NP-hard.

Follows from Theorem 7 and Theorem 3.2 of [3]. \square

Due to this issue with achieving a good approximation of the counting version, in this paper we shall focus only on determining a single optimal solution to a GBGOP - rather than all solutions.

IV. INTEGER PROGRAMS FOR SOLVING GOPs

In this section, we present an integer programming (IP) algorithms for both GBGOP and BMGOP which provide exact solutions. Given a GBGOP, the IP associates an integer-valued variable X_i with each action-point pair $(a_i, p_i) \in \mathcal{A} \times \mathcal{M}$ where $a_i(p_i) \cap \Theta_{out} = \emptyset$. Intuitively, $X_i = 1$ denotes that action a_i is performed at point p_i .

Definition 4.1 (GBGOP-IP): Let set $R = \{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}$. For each action-point pair $(a_i, p_i) \in R$, create variable $X_i \in \{0, 1\}$.

$$\min \sum_{i=1}^{|R|} X_i \quad (1)$$

$$s.t. \quad \sum_{a_j(p_j) | A_i \in a_j(p_j)} X_j \geq 1 \quad \forall A_i \in \Theta_{in} - s_0 \quad (2)$$

$$\sum_{(a_i, p_i) \in R} c_i \cdot X_i \leq \mathbf{c} \quad (3)$$

$$\sum_{(a_i, p_i) \in \Phi} X_i \leq 1 \quad \forall (\Phi \leftrightarrow \chi) \in IC_{s_0} \quad (4)$$

The objective function minimizes the total number of action-point pairs. Constraint (2) ensures that every ground atom in Θ_{in} (that does not appear in the initial state) is caused by at least one of the selected action-point pairs. Constraint (3) enforces the constraint on cost. Constraint (4) ensures that the integrity constraints are satisfied. Next we present our integer constraints for a BMGOP where the IP associates an integer-valued variable X_i with each action-point pair $(a_i, p_i) \in \mathcal{A} \times \mathcal{M}$, and an integer-valued variable Y_j with each ground atom $A_j \in B_{\mathcal{L}} - s_0$. The intuition for the X_i variables is the same as in GBGOP-IP.

Definition 4.2 (BMGOP-IP): For each action-point pair $(a_i, p_i) \in \mathcal{A} \times \mathcal{M}$, create variable $X_i \in \{0, 1\}$. For each $A_i \in B_{\mathcal{L}} - s_0$ create variable $Y_i \in \{0, 1\}$.

$$\max \sum_{A_i \in s_0} b_i + \sum_{i=1}^{|B_{\mathcal{L}}| - |s_0|} b_i \cdot Y_i \quad (5)$$

$$s.t. \quad \sum_{a_j(p_j) | A_i \in a_j(p_j)} X_j \geq Y_i \quad \forall A_i \in B_{\mathcal{L}} - s_0 \quad (6)$$

$$\sum_{(a_i, p_i) \in \mathcal{A} \times \mathcal{M}} X_i \leq k \quad (7)$$

$$\sum_{(a_i, p_i) \in \mathcal{A} \times \mathcal{M}} c_i \cdot X_i \leq \mathbf{c} \quad (8)$$

$$\sum_{(a_i, p_i) \in \Phi} X_i \leq 1 \quad \forall (\Phi \leftrightarrow \chi) \in IC_{s_0} \quad (9)$$

In the above IP, the objective function looks at each ground atom and sums the associated benefit if the associated Y_i variable is 1 - meaning that atom A_i is true after the actions are applied. Constraint (6) effectively sets a Y_i variable to 1 if an action that causes A_i to be true occurs. Constraint (7) enforces the cardinality requirement. Constraints 8-9 mirror constraints 3-4 of GBGOP-IP. The result below shows that a solution σ to the above IPs⁵, when restricted to the X_i variables, provides an immediate solution to the GOP.

Prop. 4.1: Suppose Γ is a GBGOP (resp. BMGOP) and $IP(\Gamma)$ is its corresponding integer program (GBGOP-IP, resp. BMGOP-IP). Then:

- 1) If SOL is a solution to Γ , then there is a solution σ of $IP(\Gamma)$ such that $\sigma \supseteq \{X_i = 1 \mid (a_i, p_i) \in SOL\}$.
- 2) If σ is a solution to $IP(\Gamma)$, then there is a solution SOL to Γ such that $\{X_i = 1 \mid (a_i, p_i) \in SOL\} \subseteq \sigma$.

As integer programming is NP-complete, any algorithm to solve a GOP using GBGOP-IP or BMGOP-IP using an IP solver will take exponential time. We note that for GBGOP-IP, the number of variables is fairly large - $O(|\{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}|)$ variables and $O(|\Theta_{in} - s_0| + |IC_{s_0}| + 1)$ constraints. BMGOP-IP has even more variables - (though not exponential) - $O(|\mathcal{M}| \cdot (|\mathcal{A}| + |\mathcal{G}|))$ variables and $O(|\mathcal{M}| \cdot |\mathcal{G}| + |IC_{s_0}| + 2)$ constraints. However, BMGOP-IP has only packing constraints.⁶ We also note the GBGOP-IP has both covering (\geq) and packing (\leq) constraints - another source of complexity.

V. CORRECT VARIABLE REDUCTION FOR GBGOP-IP

The set of integer constraints for GBGOP has $O(|R|)$ variables where $R \subseteq \mathcal{A} \times \mathcal{M}$. We show how to correctly reduce the number of variables by considering only a subset of R - thereby providing a smaller integer program. Our intuition is that an optimal solution SOL is an *irredundant* cover of Θ_{in} meaning there is no subset $SOL' \subset SOL$ that is also a solution. Hence, we can discard certain elements

⁵A solution to GBGOP-IP or BMGOP-IP is an assignment of values to variables that optimizes the objective function. Thus, a solution can be described as a set of equations assigning values to the variables X_i, Y_j .

⁶It is trivial to eliminate constraint 6 and re-write 5 as a non-linear objective function.

of R that cannot possibly be in an optimal solution. First, for a given GBGOP $\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathbf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, we introduce $Q_{(a,p)}^\Gamma = \{\Phi | (\Phi \leftrightarrow \chi) \in IC_{s_0} \wedge (a,p) \in \Phi\}$ and the set of ground atoms each action-point pair affects $\text{Aff}_{(a,p)}^\Gamma = a_i(p_i) \cap (\Theta_{in} - (\Theta_{in} \cap s_0))$. We can now define a *reduced action-point set*.

Definition 5.1 (Reduced Action-Point Set): Given GBGOP $\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathbf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$ and set $R = \{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}$, we define **reduced action-point set** $R^* = \{(a_i, p_i) \in R | \nexists (a_j, p_j) \in R \text{ s.t. } (c_j \leq c_i) \wedge (Q_{(a_j,p_j)}^\Gamma \subseteq Q_{(a_i,p_i)}^\Gamma) \wedge (\text{Aff}_{(a_i,p_i)}^\Gamma \subseteq \text{Aff}_{(a_j,p_j)}^\Gamma)\}$

Example 5.1: Consider the campaign scenario last discussed in Example 2.5. Suppose the candidate wants to optimize the following GBGOP: $\Gamma = (\mathcal{M}_{cpgn}, s_{cpgn}, \mathcal{A}_{cpgn}, \mathbf{C}_{cpgn}^{(s_{cpgn})}, IC_{cpgn}, 4, \Theta_{in}^{cpgn}, \emptyset)$ where each $A \in \Theta_{in}^{cpgn}$ has the form *exposure*(p) where p is a point in one of the two dashed rectangles in Figure 1. Note that as map \mathcal{M}_{cpgn} contains 187 points, $|\mathcal{A}| = 3$, and $\Theta_{out} = \emptyset$, the cardinality of R is 561. By contrast, the set R^* consists of only 7 elements, 1.2% of the size of R . Here $R^* = \{(nor, (5, 4)), (nor, (5, 3)), (nor, (5, 2)), (nor, (10, 8)), (nor, (10, 7)), (nor, (10, 6)), (appeal_1, (4, 3))\}$

Intuitively, all elements in R^* are preferable for membership in an optimal solution over $R - R^*$ as they cost less, result in the same changes to the state, and occur in the same or fewer integrity constraints. Set R^* can be found in quadratic time with a naive algorithm - an operation that is likely dominated by solving or approximating GBGOP-IP. The next lemma says that R^* must contain an optimal solution. any optimal solution to a GBGOP. This can then be used to correctly reduce the number of variables in GBGOP-IP.

Lemma 5.1: Given GBGOP $\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathbf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, for any optimal solution $SOL \subseteq R$, there is an optimal solution $SOL' \subseteq R^*$.⁷

Prop. 5.1: Suppose Γ is a GBGOP and $IP(\Gamma)$ is its corresponding integer program. We can create such a program with a variable for every element of R^* (instead of R) and Proposition 4.1 still holds true.

VI. THE BMGOP-COMPUTE ALGORITHM

While BMGOP-IP can solve a BMGOP exactly, doing so is computationally intractable. We now present an approximation algorithm that runs in PTIME but provides a lower approximation ratio than proved in Theorem 4. First, we show that a BMGOP reduces to an instance of submodular maximization problem⁸ with respect to packing constraints. We then leverage some known methods [4] to solve such problems and develop a fast, deterministic algorithm to approximate BMGOP with an approximation bounds. Given BMGOP $\Gamma = (\mathcal{M}, s_0, \mathbf{B}, \mathcal{A}, \mathbf{C}, IC, k, \mathbf{c})$, consider the objective function in BMGOP-IP. We can write that function as

⁷*Proof Sketch.* We show this by proving that for any set $W = SOL \cap (R - R^*)$, there is some set $W' \subseteq R^* - (R^* \cap SOL)$ s.t. $(SOL - W) \cup W'$ is also a solution.

⁸Suppose Z is a set. A function $f : 2^Z \rightarrow \mathbb{R}$ is said to be *submodular* iff for all Z_1, Z_2 such that $Z_1 \subseteq Z_2$ and all $z \notin Z_2$, it is the case that $f(Z_1 \cup \{z\}) - f(Z_1) \geq f(Z_2 \cup \{z\}) - f(Z_2)$, i.e. the incremental value of adding z to the smaller set Z_1 exceeds the incremental value of adding it to the larger set Z_2 . Here, \mathbb{R} denotes the reals.

a mapping from action-point pairs to reals. We denote this function (specific for BMGOP Γ) as $f_\Gamma : 2^{\mathcal{A} \times \mathcal{M}} \rightarrow \mathbb{R}^+$, where $f_\Gamma(S) = \sum_{A_i \in \text{appl}(S, s_0)} b_i$, which has certain properties.

$$f_\Gamma(S) = \sum_{A_i \in \text{appl}(S, s_0)} b_i \quad (10)$$

We now show that this function f_Γ is submodular and has some other nice properties as well.

Prop. 6.1: For BMGOP Γ , function f_Γ is: (i) submodular, (ii) monotonic, i.e. $Z_1 \subseteq Z_2 \rightarrow f_\Gamma(Z_1) \leq f_\Gamma(Z_2)$ and (iii) under the condition $\forall A_i \in B_{\mathcal{L}}, b_i = 0$, we have $f_\Gamma(\emptyset) = 0$.⁹

Proof Sketch. Consider $S \subseteq S' \subseteq \mathcal{A} \times \mathcal{M}$ and $(a, p) \notin S'$. We must show $f_\Gamma(S \cup \{(a, p)\}) - f_\Gamma(S) \geq f_\Gamma(S' \cup \{(a, p)\}) - f_\Gamma(S')$. Suppose, BWOC $f_\Gamma(S \cup \{(a, p)\}) - f_\Gamma(S) < f_\Gamma(S' \cup \{(a, p)\}) - f_\Gamma(S')$. Then, by Equation 10, we have $\sum_{A_i \in \text{appl}(S \cup \{(a, p)\}, s_0) - \text{appl}(S, s_0)} b_i < \sum_{A_i \in \text{appl}(S' \cup \{(a, p)\}, s_0) - \text{appl}(S', s_0)} b_i$. However, by the definition of *appl*, we have $\text{appl}(S \cup \{(a, p)\}, s_0) - \text{appl}(S, s_0) \supseteq \text{appl}(S' \cup \{(a, p)\}, s_0) - \text{appl}(S', s_0)$, which is a contradiction. \square

As our objective function is submodular, and constraints 7-9 are linear packing constraints, any instance of a BMGOP can be viewed as maximization of a submodular function wrt linear packing constraints and hence, methods to solve such problems can be used here. The BMGOP-Compute algorithm leverages this idea and illustrated in Example 6.1.

BMGOP-Compute

INPUT: BMGOP $(\mathcal{M}, s_0, \mathbf{B}, \mathcal{A}, \mathbf{C}, IC, k, \mathbf{c})$

OUTPUT: $SOL \subseteq \mathcal{A} \times \mathcal{M}$

- 1) Set $SOL = \emptyset$, δ to be an infinitesimal, and set $\lambda = e^{2-\delta} \cdot (2 + |IC_{s_0}|)$.
 - 2) Set $w' = 1/k$ and $w'' = 1/\mathbf{c}$. For each $(\Phi_i \leftrightarrow \chi_i) \in IC_{s_0}$, set $w_i = 1/(2 - \delta)$.
 - 3) While $k \cdot w' + \mathbf{c} \cdot w'' + (2 - \delta) \cdot \sum_i w_i \leq \lambda$ and $SOL \neq \mathcal{A} \times \mathcal{M}$
 - a) Let $(a_j, p_j) \in \mathcal{A} \times \mathcal{M} - SOL$ have minimal $\frac{w' + w'' \cdot c_j + \sum_i (a_j, p_j) \in \Phi_i w_i}{(\sum_{A_i \in \text{appl}(SOL \cup \{(a_j, p_j)\}, s_0)} b_i) - (\sum_{A_i \in \text{appl}(SOL, s_0)} b_i)}$
 - b) $SOL = SOL \cup \{(a_j, p_j)\}$
 - c) Set $w' = w' \cdot \lambda^{1/k}$, $w'' = w'' \cdot \lambda^{c_j/\mathbf{c}}$ and for each integrity constraint i s.t. $(a_j, p_j) \in \Phi_i$, set $w_i = w_i \cdot \lambda^{1/(2-\delta)}$
 - 4) If SOL is not a valid solution then
 - a) If $\sum_{A_i \in \text{appl}(SOL - \{(a_j, p_j)\}, s_0)} b_i \geq \sum_{A_i \in \text{appl}(\{(a_j, p_j)\}, s_0)} b_i$, then $SOL = SOL - \{(a_j, p_j)\}$
 - b) Else $SOL = \{(a_j, p_j)\}$
 - 5) Return SOL
-

Example 6.1: Following Example 2.5. Suppose the candidate wants to optimize BMGOP: $(\mathcal{M}_{cpgn}, s_{cpgn}, \mathbf{B}_{cpgn}, \mathcal{A}_{cpgn}, \mathbf{C}_{cpgn}^{(s_{cpgn})}, IC_{cpgn}, 3, 2)$. In this case, we will set $\delta = 0.001$. He wishes to find a set of 3 action-point pairs to optimize his exposure. BMGOP-Compute sets $\lambda = 22.14$, $w' = 0.33$, $w'' = 0.50$, and

⁹Henceforth, we will assume this condition to be true.

$w_1 = 0.50$ in lines 1 and 2. In the first iteration of the loop at line 3, it finds the action-point pair that minimizes the quantity at line 3 is $(appeal_1, (4, 3))$ - which has the associated value 0.073. Note, other action-point pairs with low values are $(appeal_2, (10, 7))$ with 0.083 and $(nor, (15, 6))$ also with 0.083. It then adds $(appeal_1, (4, 3))$ to SOL and updates $w' = 0.93$, $w'' = 1.09$, and $w_1 = 2.35$. On the next iteration, the **BMGOP-Compute** picks $(nor, (15, 6))$, which now has a value of 0.164. During this iteration, the value of $(appeal_2, (10, 7))$ has increased substantially - to 0.294, so it is not selected. At the end of the iteration, w' is updated to 2.611 and w'' is updated to 2.364. As $(nor, (15, 6))$ does not impact the lone integrity constraint, the value w_1 remains at 2.354. In the third iteration, **BMGOP-Compute** selects $(nor, (15, 9))$ which has a value of 0.421. Again, the value of $(appeal_2, (10, 7))$ has increased - but this time only to 0.472. **BMGOP-Compute** re-calculates $w' = 7.331$, $w'' = 5.128$ and w_1 remains at 2.354. On the last iteration, **BMGOP-Compute** picks $(appeal_2, (10, 7))$ as it has the lowest value - 0.942. After this fourth iteration, it updates $w' = 20.589$, $w'' = 11.124$, and $w_1 = 11.0861$ - which now total to 42.799 - exceeding λ (22.14) - causing **BMGOP-Compute** to exit the outer loop. Now SOL has 4 elements, exceeding the cardinality constraint (as well as the integrity constraint). The checks done in line 4 remove $(appeal_2, (10, 7))$ from SOL - making the result feasible. **BMGOP-Compute** returns $\{(appeal_1, (4, 3)), (nor, (15, 6)), (nor, (15, 9))\}$ which causes the benefit to be 45.

Prop. 6.2: Suppose Γ is a BMGOP and SOL is the set returned by **BMGOP-Compute**. Then SOL is a solution to Γ .¹⁰

Next, we show **BMGOP-Compute** runs in PTIME.

Prop. 6.3: **BMGOP-Compute** runs in $O(k \cdot |\mathcal{M}| \cdot |\mathcal{A}| \cdot |IC_{s_0}|)$ time.

Proof Sketch. Clearly, the outer loop can iterate no more than k times. The inner loop iterates for each element of $\mathcal{A} \times \mathcal{M}$ - hence requiring time $O(|\mathcal{M}| \cdot |\mathcal{A}|)$. There are some additional operations that require $O(|IC_{s_0}|)$ time, however, they are dominated under the assumption that $|\mathcal{M}| \cdot |\mathcal{A}| \gg |IC_{s_0}|$, which we expect in our application. \square

The following important theorem states that **BMGOP-Compute** provides an approximation guarantee. Because of Theorem 4 and as **BMGOP-Compute** is polynomial, we know that this approximation guarantee cannot be as good as $\frac{e-1}{e} + \epsilon$. The result leverages Theorem 1.1 of [4] together with the above theorems. By this result, the approximation factor of **BMGOP-Compute** depends on $|IC_{s_0}|$. We illustrate this relationship, in Figure 2. For our target applications, we envision $|IC_{s_0}| \leq 20$.

Theorem 9: Under the assumption that $k, \mathbf{c} \geq 2 - \delta$, **BMGOP-Compute** provides a solution within a factor of $\frac{1}{(2+|IC_{s_0}|)^{1/(2-\delta)}}$ (where δ is an infinitesimal) of optimal.

Proof Sketch. **BMGOP-Compute** follows from Algorithm 1

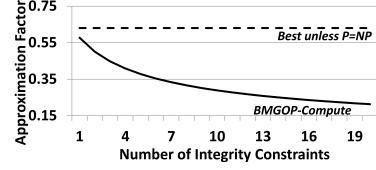


Fig. 2. $|IC_{s_0}|$ vs. approximation ratio.

of [4] which optimizes a submodular function subject to m packing constraints within $\frac{1}{m^{1/W}}$ where W is the minimum width of the packing constraints - defined as the minimum of the size of the constraint divided by the cost of an element. For constraint 7, the $W = k$. For constraint 8, the $W \geq \mathbf{c}$. We can replace constraint 9 with: $\sum_{(a_i, p_i) \in \Phi_j} X_i \leq 2 - \delta \forall (\Phi_j \leftarrow \chi_j) \in IC_{s_0}$ which maintains correctness as two variables to set to 1 and exceeds $2 - \delta$. The new constraint has width $2 - \delta$, which is the minimum. We then apply Theorem 1.1 of [4]. \square

Discussion. We note that while a BMGOP reduces to the maximization of a submodular or linear function wrt linear packing constraints, there are other algorithms available besides the multiplicative update algorithm of [4]. However, we feel that this is likely the best approach for several reasons that we list below.

- 1) The approximation ratio achieved by the multiplicative-update algorithm matches the best approximation ratio achievable for maximizing a linear function wrt linear packing constraints (see [5]), hence, it is unlikely that a better approximation ratio can be achieved using such a technique.
- 2) Other methods (such as those presented in [5]) require solving a relaxation of the associated MILP. In our case, such an operation would take $O((|\mathcal{M}| \cdot (|\mathcal{A}| + |\mathcal{G}|))^{3.5})$ time (as a consequence of the number of variables in **BMGOP-MILP** and the results of [6]). This is significantly more expensive than the $O(k \cdot |\mathcal{M}| \cdot |\mathcal{A}|)$ of **BMGOP-MU** (see Proposition 6.3). If the map, \mathcal{M} is very large, solving a relaxation of **BMGOP-MILP** may be unrealistic on most hardware.
- 3) The algorithm **BMGOP-MU** is totally deterministic, which allow us to avoid the issue de-randomization.
- 4) The algorithm **BMGOP-MU** is guaranteed to provide a solution that meets constraints 7-9 - as opposed to only meeting them probabilistically.

VII. RELATED WORK AND CONCLUSIONS

Though spatial reasoning has been studied extensively in AI [7], [8], [9], [10], many of the paradigms that have emerged for such reasoning are *qualitative* in nature. Such qualitative spatial reasoning efforts include the influential region connection calculus for qualitative reasoning about space. There has also been work on quantitative methods for reasoning about space [11] which contains articles on spatial reasoning in the presence of uncertainty using both logical and fuzzy

¹⁰Here, SOL is not necessarily an optimal solution.

methods. Spatial reasoning with quantitative information has been studied extensively in image processing [12], [13].

However, unlike this vast body of work, this paper focuses on a different problem. Suppose we are dealing with a map \mathcal{M} , a cost function \mathbf{C} , a set \mathcal{A} of possible actions, a bound on the cost \mathbf{c} , and a bound on the number of actions we can take, what set of actions should be taken so as to optimize a given objective function. Two versions of this problem are studied in this paper - GBGOP and BMGOP which differ in what they optimize. Both problems are proved to be NP-hard (NP-complete under realistic assumptions) and we further prove that the number of solutions to GBGOP is #P-complete. We also find limits on approximating an optimal solution to BMGOP and GBGOP (in PTIME) under accepted theoretical assumptions. We develop integer programming formulations of both problems and then present a way of simplifying the IP for GBGOP. We further present the BMGOP-Compute algorithm for BMGOP and show that it is polynomial and has a guaranteed approximation ratio (though not high enough to contract the NP-hardness result).

VIII. CONCLUSION

In this paper, we introduced “geospatial optimization problems” or GOPs that aide the user in taking certain actions over a geographic region. We showed these problems to be NP-hard and provided integer constraints. For the goal-based variant, we correctly reduce the number of variables. For the benefit-maximizing variant, we provide an approximation algorithm. In future work, we look to implement this framework and explore methods to achieve further scalability, as well as utilize geo-located social network data to establish relationships among locations in order to better implement action-point pairs and integrity constraints.

REFERENCES

- [1] T. Eiter, V. Subrahmanian, and G. Pick, “Heterogeneous Active Agents, I: Semantics,” *Artificial Intelligence Journal*, vol. 108, no. 1-2, pp. 179–255, 1999.
- [2] U. Feige, “A threshold of $\ln n$ for approximating set cover,” *J. ACM*, vol. 45, no. 4, pp. 634–652, 1998.
- [3] D. Roth, “On the hardness of approximate reasoning,” *Artificial Intelligence*, vol. 82, pp. 273–302, 1996.
- [4] Y. Azar and I. Gamzu, “Efficient submodular function maximization under linear packing constraints,” (submitted, preprint available from <http://www.cs.tau.ac.il/~iftgam/papers/SubmodularPacking.pdf>), 2010.
- [5] A. Srinivasan, “Improved approximation guarantees for packing and covering integer programs,” *SIAM J. Comput.*, vol. 29, pp. 648–670, 1995.
- [6] N. Karmarkar, “A new polynomial-time algorithm for linear programming,” *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.
- [7] S. M. H. Anthony G. Cohn, “Qualitative spatial representation and reasoning: An overview,” *Fundam. Inform.*, vol. 46, no. 1–2, pp. 1–29, 2001.
- [8] O. G. M. J. Egenhofer and H.-J. Schek, “Reasoning about binary topological relations,” in *Advances in Spatial Databases, Lecture Notes in Computer Science*, vol. 525, 1991, pp. 143 – 160.
- [9] J. Renz and B. Nebel, “On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus,” *Artif. Intell.*, vol. 108, pp. 69 – 123, 1999.
- [10] S. Li and M. Ying, “Region connection calculus: Its models and composition table,” *Artif. Intell.*, vol. 145, pp. 121 – 146, 2003.
- [11] R. Jeansoulin, O. Papini, H. Prade, and S. Schockaert, *Methods for Handling Imperfect Spatial Information*, ser. Studies in Fuzziness and Soft Computing, R. Jeansoulin, O. Papini, H. Prade, and S. Schockaert, Eds. Springer Berlin / Heidelberg, 2010, vol. 256.
- [12] Y. Weiss and E. Adelson, “A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models,” in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR ’96, 1996 IEEE Computer Society Conference on*, Jun. 1996, pp. 321 –326.
- [13] R. Srihari, “Automatic indexing and content-based retrieval of captioned images,” *Computer*, vol. 28, no. 9, pp. 49 –56, Sep. 1995.